THE 3D DCT (Discrete Cosine Transform):
A GENERALIZATION OF THE 2D DCT

par:
Michel J. DUMONTIER
Patrick GERLIER : *Ingénieur EP*
Dr Pascal LERAY *Image Processing & Computer Graphics Expert*
FRANCE TELECOM/CNET/DIH

Introduction:
The standard, classic, and well known 2D DCT is largely used in the MPEG or JPEG world, for very efficient image con
But the spatio-temporal 3D DCT (2 Dimensions plus time) has not yet been tested.
The present paper is dedicated to an APL implementation of the 3D DCT, defined as a generalization of the 2D DCT.

DESCRIPTION:
In this case, instead of 8*8 image blocs, we choose 8*8*8 spatio-temporal blocs.
A direct 3D DCT transforms the input bloc in the Fourier space.
A rounded vector B is obtained from this result.

An inverse 3D DCT decompression is obtained by the reverse process.
The result bloc is compared with the original bloc V13D.
The detailed implementation is shown here:

APL implementation:

The 3D DCT transform function:
```
VT←DCT3D V
⎕IO←1
N←3 1 2
VT←0.015625×MCOS+.×N⍉(MCOS+.×N⍉(MCOS+.×N⍉V))
```

The reverse 3D DCT transform function:
```
VT←IDCT3D V
⎕IO←1
N←3 1 2
VT←(⍉MCOS)+.×N⍉((⍉MCOS)+.×N⍉((⍉MCOS)+.×N⍉V))
```

Initialization function: COEFFCOS:

```
COEFFCOS
⎕IO←0
FAC←1
PI←3.141593653589
A←FAC×cos(PI÷4)
B←FAC×cos(PI÷16)
C←FAC×cos(3×PI÷16)
D←FAC×cos(5×PI÷16)
E←FAC×cos(7×PI÷16)
F←FAC×cos(PI÷8)
G←FAC×cos(3×PI÷8)
```

```
(PI÷16)∘.×⍳8
MCOS←8 8⍴0
MCOS[0;]←8 1⍴A,A,A,A,A,A,A,A
MCOS[1;]←8 1⍴B,C,D,E,(-E),(-D),(-C),(-B)
MCOS[2;]←8 1⍴F,G,(-G),(-F),(-F),(-G),G,F
MCOS[3;]←8 1⍴C,(-E),(-B),(-D),D,B,E,(-C)
MCOS[4;]←8 1⍴A,(-A),(-A),A,A,(-A),(-A),A
MCOS[5;]←8 1⍴D,(-B),E,C,(-C),(-E),B,(-D)
MCOS[6;]←8 1⍴G,(-F),F,(-G),(-G),F,(-F),G
MCOS[7;]←8 1⍴E,(-D),C,(-B),B,(-C),D,(-E)
```

GLOBAL TEST FUNCTION: TST3D
On 8*8*8 Spatio-temporal blocs:

```
Z←TST3D
⎕IO←1
COEFFCOS
A←DCT3D V13D
B←0.01×⌊A×100
Z←IDCT3D B
Z←'F6.1'⎕FMT Z[1;;]
```

The cosine function COS:

```
z←cos v
z←2○v
```

V13D is a 8*8*8 spatio-temporal bloc
V13D is loaded with a IOTA N coefficient

The result shows that the reconstructed V13D value is equivalent to the initial value.
V13D: 8 times this bloc:

```
 1  2  3  4  5  6  7  8
 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64
```

The transformed matrix: B: (B is rounded from A)

```
 91.92 ¯6.45 0 ¯0.68 0 ¯0.21 0 ¯0.06
¯51.54  0    0  0    0  0    0  0
  0     0    0  0    0  0    0  0
 ¯5.39  0    0  0    0  0    0  0
  0     0    0  0    0  0    0  0
 ¯1.61  0    0  0    0  0    0  0
  0     0    0  0    0  0    0  0
 ¯0.41  0    0  0    0  0    0  0

 ¯0.01  0    0  0    0  0    0  0
  0     0    0  0    0  0    0  0
  0     0    0  0    0  0    0  0
```

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Conclusion:

Only 9 coefficients are different from 0.
All the energy is concentrated on the first coefficients.
(As for the classic 2D DCT transform)
But in the 3D case, the compression ratio is much more efficient:

Compression ratio possible evaluation:
2 coefficients using: 16bits
1 coefficients using: 10bits
5 coefficients using 8 bits

Total: 34 bits (< 9 octets, for 8*8*8 = 512 coefficients of 16 bits)
For this example, the compression ratio is:
1024/9 = 113

FURTHER RESEARCHES:

Several other examples could be tested, mainly for real images:
One could transform an animated spatio-temporal sequence, and test with the 3D DCT transform.
Test sequences could be choosen among images with low or high frequencies.
(classic DCT beeing more efficient for low frequencies)
But adaptive quantizing could be used, with, for ewample, neural nets in order to optimized the quantizing process.

GENERAL CONCLUSION:

This first result seem very promising.
Further implementations and links with real image sequences must be done and tested now.

But at present, APL implementation demonstrates a very efficient coding compacity, able to solve very complex algorith short time, without the complex requirements of C or C++.